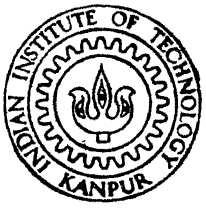
A SOFTWARE ENVIRONMENT FOR ANALYSIS AND SIMULATION OF DSP SYSTEMS

by

Capt. S. K. TRIPATHI



DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

FEBRUARY, 1991

A SOFTWARE ENVIRONMENT FOR ANALYSIS AND SIMULATION OF DSP SYSTEMS

A Thesis Submitted

In Partial Fulfilment of the Requirement

for the Degree of

MASTER OF TECHNOLOGY

by

Capt. S. K. Tripathi

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

FEBRUARY, 1991

CERTIFICATE

It is certified that the work contained in the thesis entitled ' A Software Environment For Analysis And Simulation Of DSP Systems ' by Capt. S. K. Tripathi has been carried out under my supervision and that the work has not been submitted elsewhere for a degree.

Anil Mahanta

(Dr. Anil Mahanta)
Assistant Professor
Department Of Electrical Engineering
Indian Institute Of Technology
KANPUR

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to Dr. Anil Mahanta, my thesis supervisor, for his excellent guidance, encouragement and above all his considerate approach.

I am grateful to Mr. A.C. Trivedi, research engineer, for his help in trouble shooting .

I thank Mr A. C. Joshi for providing excellant repair support to keep my PC in working condition all the time.

I thank Sri S.K. Tewari for his help and guidance during printing of thesis work.

Capt. S. K. Tripathi

TO

Little Richa

ABSTRACT

A software environment for analysis and simulation of DSP systems is presented. This is a software which offers facilities for signal processing on digital data and provision for simulation and testing of a DSP system.

User can generate data sequences from software library or read disk files, process or manipulate data sequences in desired form, display the result by graph-plot and store it in software buffers or files for later use. User can also define a DSP system in a number of standard ways to study its behaviour in time and frequency domain. Input quantisation and coeffecients quantisation effects can also be studied. The facility to simulate a DSP system in terms of block diagram flow graph is provided for studying and experimenting with types of systems and its parameters.

The software has been developed for an IBM compatible PC with a hard disk and a Hercules/CGA/EGA graphics card. Turbo Pascal version 5 has been used to write the software with graphics support of Turbo Pascal Graphics Toolbox.

CONTENTS

CHAPTER-1

INTRODUCTION

Over the last few years, a number of software packages have been developed for analysis, design and simulation of DSP and communication systems. Analylatical techniques for performance evaluation of a complex DSP or Communication system may not always be feasible except in some simple and idealised situations. On the other hand, software simulation of a system not only overcomes such limitations but also offers the advantage of not getting involved in the hardware development which can be cumbersome, expensive, time consuming and inflexible. Further, in the context of learning DSP, a student may not always successfully grasp the key concepts by study of the theory alone whereas a DSP environment created for the student enables him to experiment and thereby understand the concepts in a much more direct way.

The simulation environment has changed considerably in the last 20 years. In the late 60's and early 70's simulation usually involved main frame computers operating in a batch processing mode with a higher level language being used. During the last few years, microcomputers have become powerful and inexpensive, enough so that it is quite common to have significant computing power on a desk-top. For many applications, a microcomputer such as IBM-PC provides sufficient computing power for accomplishing design evaluation and accurate analysis of a DSP system. Since the major advantage with the PC is that it is not a shared resource and, therefore, the user has complete control over simulation

environment and also long run times are not a problem. Further, such an easily available resource eacourages experimentation with the system under study. A simulation system can be of two types :

(a) Use of conventional programming language either to provide a special library of functions along with an interface or to solve a specific signal processing problem of interest.

(b) To create a programming environment for DSP wherein User can define the system in terms of library blocks and to have flexibility of easy modification of system parameters and structure.

There are a number of libraries available for signal processing which provide code to perform many signal processing operations in a user's program. The user still has to do some programming unless an interactive interface is also provided to handle the subroutines available in the libraries.

1.1 DSP SYSTEMS AVAILABLE OR UNDER DEVELOPMENT

A number of DSP simulation systems have been developed and quite a few are being developed. Some of these systems are being discussed in the following paragraphs. Systems numbered from 1 to 5 can be classified as of the type (a) and 6 and 7 as of the type (b) mentioned earlier.

1. SYSTID . [1] It is a software package of communication and signal processing system simulation, analysis and design. The user interface is highly structured and flexible and generates block diagram outputs of simulation problem topology. SYSTID is written in Fortran.

2. MODEM . [2] It is a simulation program written in Pascal under the DOS operating system. The software is completely menu driven and allows user developed models to be integrated into the simulation package. This requires an IBM-PC with a math coprocessor and CGA graphics card.

3. MATLAB . [3] It provides specialised signal processing environment. MATLAB includes a large number of routines for linear algebra and signal processing. These routines can be used interactively or combined by the user into new functions.

4. MATHMATICA . [4] It provides many of the desirable characteristics of a system for research in signal processing. It is a tool for symbolic mathematics, interactive modelling and is a powerful aid for for learning.

5. SPDTEK . [5] This package has been developed by Tektronix. It provides a wide range of capabilities including waveform acquisition, waveform generation, signal processing, mathematical functions, waveform graphics, waveform manipulation and waveform file I/O. The package has been written in C language.

6. SRL and SPLICE . [6] These subroutine libraries provide a programming environment for signal processing. They use LISP programming language and use object oriented techniques.

7. QUICKSIG. [6] It provides an experimental object oriented DSP programming environment. The system is used extensively in speech processing studies and is very useful in introducing basic DSP concepts to the students. It is written using LISP language.

1.2 DSP SOFTWARE ENVIRONMENT FOR REAL TIME IMPLEMENTATION

The easy availability of low cost and high performance DSP microprocessors has brought in another dimension in the simulation scenario. The simulation software can generate codes for such processors and also schedule the processing activity. This in effect actually implements the system in real time, at the same time allowing the flexibility offered by the software. Some of the systems aimed at real time implementation are discussed below.

1. DSPLAY . [7] It is a block diagram system for DSP, developed by Burr Brown Corporation. It is a PC based system that generates codes for AT&T DSP-32. In DSPLAY, DSP sytems are built graphically as block diagrams using higher level signal processing functions. It requires an IBM compatible PC with a CGA or EGA card.

2. GABRIEL . [8] This is a second generation design environment developed at the University of California, Berkeley. This simulation program uses block diagrams and a very flexible model to support multirate and asynchronous systems. This project was started in 1985 with an object to apply some of the principles to real time systems. The software part is still under enhancement. Gabriel system can be defined by block diagrams, with blocks from the standard library or generated by the user. Gabriel generates assembly codes for one or more digital signal processors to implement the system in real time. The language used is LISP.

1.3 AIM OF THE THESIS AND ITS ORGANISATION

In this thesis, we have attempted to develop a fully interactive software with the following objectives in mind.

(a) To provide a DSP environment for study of DSP concepts, experimenting with DSP systems and processing of digital data.

(b) Simulation of complex DSP systems represented by block diagram flow graphs.

An interactive software is an effective means to teach and disseminate results since it has an important characteristics that the work can be developed and stored from the same system. The user is able to explore and experiment with various facilities being provided by the software. A good interactive system prompts and guides the user in a manner so that all the paths open to the user get highlighted and thus enabling the user to decide on an option. While learning a particular DSP concept, the user can manipulate the results by changing of parameters and thus gets a better feel of the concept or experiment under study.

The software developed in this thesis offers an extensive library of subroutines with a fully interactive interface for signal processing and facilities to simulate a DSP system. The software has been written using Turbo Pascal version 5 with the graphics support of Turbo Pascal Graphics Toolbox. The software can be installed on an IBM compatible PC with a hard disk and a Hercules/CGA/EGA graphics card. This software contains two EXE files, namely DSP.Exe and SYSTEM.Exe.

The DSP.Exe file offers a menu driven interactive software system. All DSP functions are called from the main menu and after execution of the called function, user returns to the main menu. When a function is called, user is prompted for selection of input signal(s) and function parameters. Input and output of the

function are displayed by graphs and user has a choice of storing the result. The details of this software are discussed in Chapter 2.

A complex DSP system can be considered to be consisting of a number of subsystems (Blocks), with each sub-system performing a signal processing operation on digital data. In SYSTEM.Exe file, for simulation of a DSP system, each sub-system can be considered to be a functional block. The output of each block can be studied with different types of input signals and by varying parameters of the block. By extensive experimenting on a simulated DSP system, the parameters of the system can be decided to achieve the desired results, which can act as a guideline for development of the hardware and, thus, the software can be used for designing a system. The details of this software are discussed in Chapter 3.

CHAPTER - 2

DIGITAL SIGNAL PROCESSING

2.1 INTRODUCTION

The aim behind development of the DSP software (DSP.Exe, Appendix B) is to provide a Digital Signal Processing environment to the user which facilitates signal generation, various signal processing operations on digital data, study of DSP system response and display and storage of processed data. The software is designed to be fully interactive with the user for ease of operation and understanding.

2.2 EVOLUTION OF THE DSP SOFTWARE

In a DSP system, an input sequence (or a set of input sequences) is subjected to an operation (function) or a sequence of operations to produce an output sequence. After processing, user may like to view the processed signal or examine its details. She (he) may want to store the result in a disk file or in a buffer (for temporary storage). Accordingly four main blocks may be identified in a DSP system.

- Input Block for generation of various standard signals (impulse,sine etc.) or reading of input sequence from a disk file

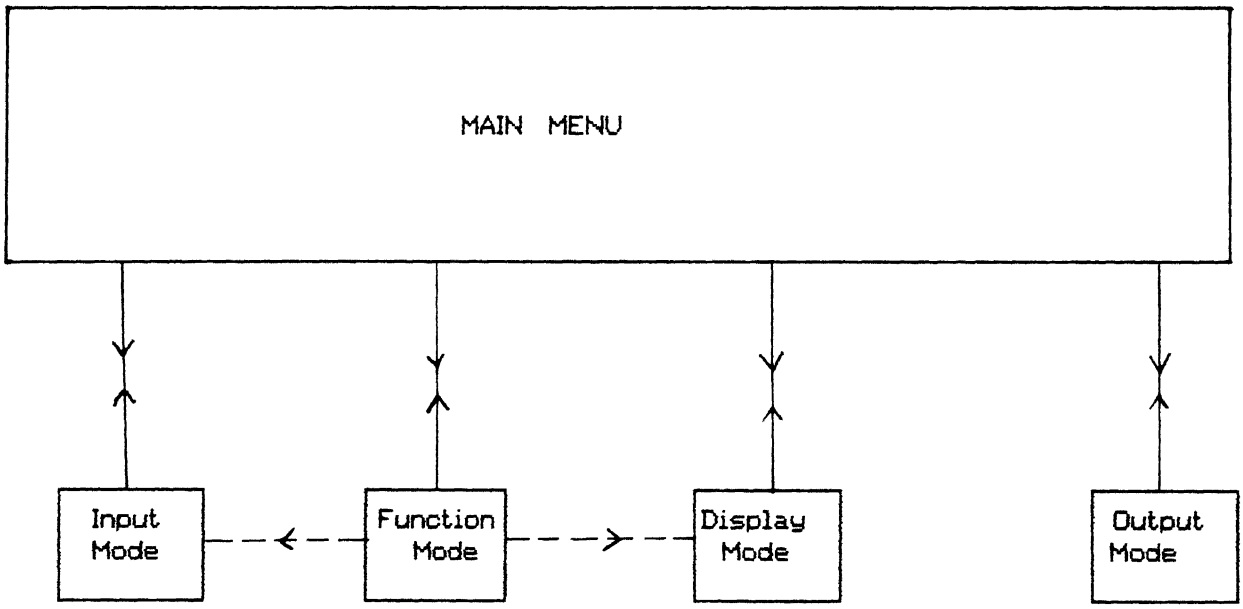
– Function block for processing of data sequence by signal processing, arithmetic and sequence conversion functions.

– Display block to facilitate visual examination and tabulation of result.

– Output block to store processed data in disk files.

The DSP software reported in this thesis has evolved exactly along these directions. The opening menu (of DSP.Exe file) displays four blocks or modes (Fig 2.1) which are accessible independently from main menu. In a typical signal processing activity, the user enters the function mode and selects a function which, in turn, brings the user to the input mode for selection of input(s). Upon completion of the processing, the output is automatically displayed by graph-plot and user is given a choice to store the results in software buffers before he is brought to the main menu again. If desired, the user may enter display mode for plotting or tabulation of results and return to main menu . Further, if results are to be stored in a file, the user enters output mode to transfer data to file and returns to main menu. At first glance it appears that the Input, Display and Output modes could very well have been organised as part of the function mode rather than being organised as independent modes accessible from main menu. This is rightly so in typical signal processing activity where the modes are invoked as a natural consequence, and having to call them by coming to main menu is slightly cumbersome. However, organising the blocks in heirarchical manner restricts the flexibility of the software. For instance, if user wants to create an input and display the data sequence, or store this result in a buffer/ disk file, independent access to modes would be more convenient rather than the heirarchial organisation. For this and other similar reasons, we decided to retain organisation given in Fig 2.1, and at the same time provide additional links between function and input modes and between function and display modes. With this modifications, (dotted line in Fig 2.1), function mode automatically invokes

FIG 2.1 - ORGANISATON OF MODES



Note :- Dotted lines represent additional links between Function and other modes.

input creation and automatically displays result in form of graph-plot. With these requirements in mind, the software has been designed to work in four basic modes, as given below.

1. Input mode . To generate standard signals or to read data from buffers / disk-files.

2. Function mode . This mode provides a facility to carry out signal processing, arithmatic and sequence conversion operations. It automatically calls input mode for selection of input type. This mode also offers a facility for study of quantisation effcts and system response.

3. Display mode . To display intermediate results stored in software buffers.

4. Output mode . To transfer data to disk files.

2.2.1 Aims of DSP Software

Broadly, the software aims to provide the following facilities.

1. Generation of input signal from standard library routines provided in the software.
2. Retrieval of data sequence from disk files or software buffer (buffers provided for temporary storage of data) and use this sequence as input signal.
3. Provision of various functions to perform following operations on a signal:
   - Signal Processing functions to carry out DSP operations (DFT, FFT, Interpolation, convolution etc.)
   - Arithmatic functions to carry out essential arithmatic operations on data sequence (Addition, Multiplication, Logarithm etc.)
   - Sequence conversion functions to manipulate data sequence to desired form (Rectagular to polar, Zero pad, Extraction etc.)
4. Provision of Quantisation function for the following:

– To quantise an input signal

– To study the effect of coeffecient quantisation on a DSP system defined in Dirct form or Cascade form

5. Provision of 'System- Response' function to study Impulse/ Step/ Frequency response of a DSP system which is defined in the following forms.

   (a) System defined in time domain by difference equation.

   (b) System defined in frequency domain as one of the following type.

      – Direct form

      – Cascade form

      – Parallel form

      – System defined by its poles and zeroes

6. Display facility to tabulate/ graph-plot intermediate results stored in a software buffer.

7. Output facilty to store data in a disk file with source of data from one of the following.

   – Generated data sequence from standard library routines provided in the software.

   – Intermediate results stored in a software buffer.

   – Data from a disk file in any drive/directory (i.e., to transfer data from one file to another file).

2.3 BUFFERS FOR DATA STORAGE

Temporary storage of intermediate data is an important facility in a signal processing software. Digital data can be assumed to be always complex since real part is always present in any intermediate result and imaginary part may also be

present depending upon the type of DSP operation being performed. To fulfil storage requirement, a buffer has to store data in complex form.

2.3.1 Buffer Details

Four buffers are provided in DSP.Exe for temporary storage of real or complex data. Each buffer has a maximum lenth of 512 points for complex data storage, i.e., corresponding real and imaginary parts of data can be stored upto a lenth of 512 points each. For the storage of a data sequence of lenth shorter than 512 points, remaining lenth of buffer is filled up with zeros. Whenever only real signal is stored in any buffer, the corresponding imaginary part lenth of 512 points is filled up with zeroes. If during a DSP operation, the output data has been converted to polar form (e.g., after Rectangular to Polar conversion), this data can be stored in a buffer as magnitude and phase part (in radians), stored in place of real and imaginary part respectively.

Each buffer memorises the type of DSP operation performed on data before its storage, type of data (rectangular/ polar) and lenth of data stored. At any time, buffer status can be seen by entering display mode.

2.4 INPUT MODE

Signal generation is a fundamental requirement of a DSP software. It is desirable that the software provides a comprehensive library of standard input sequences (deterministic as well as random) for test and simulation purpose. More importantly, it should allow selection of input from disk-file or from storage buffers for real DSP applications. Additionally, it should be convenient to generate any arbitrary input signal directly from the keyboard. All these fuctions are included the in

input mode. Briefly the facilities provided by this mode are as follows.

1. Generate any signal from standard library/ keyboard or pick up a sequence from disk-file / buffer.
2. View the type of signal generated by auto display and ascertain its correctness. This facility can also be used to view contents of a disk-file or a buffer at any time.
3. Store the generated signal in a buffer for later use.

2.4.1 Standard Library Inputs

Following types of input signals are provided.

- Impulse
- Pulse
- Sine
- Cosine
- DC
- Ramp

These sequences are useful in system response evaluation and simulation of complex DSP system (Chapter 3).

2.4.2 Method of Generation

When an input is selected, user has to specify relevant parameters, e.g., offset from origin, maximum amplitude, periodic or non periodic etc. If the signal is periodic, look-up table method is used for generation of signal. A table containing data lenth of one period is filled up initially as specified by the user and read out from beginning to end repeatedly to fill up concecutive cycles of periodic signal. An input buffer is used for filling up of input signal which is filled in a way similar to that of data storage buffers given in Sec 2.3. For periodic

signals, complete lenth (512 points) of input buffer is filled up with cycles of periodic signal. For a non periodic signal of lenth shorter than 512 points, the buffer first fills up with data values followed by zeroes for the rest of the lenth.

2.4.3 Input signal from File/Buffer

Input signal can also be generated by reading data sequence from disk-file or software buffers. For reading data from file, default directory is current directory unless some other path is specified. During reading of data from software buffers which are holding result of a DSP operation (Sec 2.3), user has a choice to extract any lenth of data upto a maximum of 512 points. If a lenth longer than that of stored data lenth is extracted, the sequence is filled up with trailing zeroes after reading data values.

2.4.4 Input signal from keyboard

An arbitrary input signal can be generated directly from the keyboard, by specifying lenth of signal and then feeding values in real or complex form. This facility is useful when the user wishes to try out a sequence which does not match any of the standard library functions.

2.4.5 Display of Generated Input signal

After generation of an input signal, it is displayed by graph-plot automatically. This enables the user to ascertain the correctness of generated signal before he uses the signal for further processing.

2.4.6 Storage of Generated Input signal

If the generated signal is correct and is to be used later, it should be stored temporarily in a buffer. In the software, four buffers are

provided for such temporary storage. These buffers cater for storage of real or imaginary signals (ref 2.3). During storage, contents of input buffer are transferred to one the four storage buffers.

2.5 FUNCTION MODE

This mode should provide a facility for all types of basic signal processing, arithmetic and data sequence conversion operations. In addition, user should be able to study effcts of quantisation and response of a DSP system defined in a number of standard ways. Before execution of a selected function, user should be prompted to select type of input signal. The display of input signal and processed output should be automatic. During display, only real part should be displayed for a real signal and real and imaginary parts should be displayed for a complex signal. After an operation is performed, user should have a choice of storing the processed data in a software buffer from where it can be read later..

2.5.1 Signal Processing Functions

1. Signal Spectrum . In many signal processing applications, the distiguishing features of signals and filters are most easily interpreted in frequency domain. To get signal spectrum, the Fourier Transform is computed in the form of DFT or FFT of a discrete time sequence. Discrete Time Fourier Transform of a signal is given as

$$X(jw) = \sum_{n} x(n) \exp(-jwn) \quad ; \quad -\infty < n < \infty \qquad - (2.1)$$

In the DFT function provided, a time domain input sequence of lenth N is converted to a frequency domain sequence of lenth N. DFT is a slow process since N^2 complex

multiplications are involved for DFT computation of a N point sequence. FFT is a faster process since only (N/2) $\log_2 N$ complex multiplications are involved. FFT has a restriction of computational lenth being only in powers of 2. To fulfil this requirement any sequence can be padded with trailing zeroes to achieve a appropriate lenth in power of 2. For FFT computation, Cooley-Turkey algorithm has been used.

DFT of a sequence can also be written in the following form.

$$X(Z) = \sum_{n=0}^{N-1} x(n)\, Z^{-n} \quad ; \quad \text{where } Z = \exp(jw) \qquad - (2.2)$$

As higher powers of Z^{-i} are computed in floating point by a computer, due to limitation of floating point precision the likely chances of calculation error become higher. To overcome this drawback, a Precision DFT has been provided which computes DFT as per following algorithm (Horner's Rule).

$$X(Z) = X(0) + Z^{-1}\,[\,X(1) + Z^{-1}\,(\,X(2) + Z^{-1}\,(\,X(3) + Z^{-1}\,(\ldots\ldots)\,)\,)\,] \qquad - (2.3)$$

By using this algorithm, the powers of Z^{-n} is restricted to Z^{-1} during computation of DFT and thus high precision result can be obtained in floating point calculation.

2. Convolution and Correlation . In an LTI system, given an input x(n), the output y(n) is

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\; h(n-k) \qquad - (2.4)$$

or the output is equal to convolution of input signal and impulse response of the system. Thus convolution function aims to provide a method to calculate output of any DSP system, with impulse response of the system available, given any input. This function can also be used to convolve any two signals. Time domain convolution is performed by straight multiplication and summations whereas fast (circular) convolution is performed by multiplying the two input signals in frequency domain and converting the result to time domain. In fast convolution, due to computational error, the result of convolving two real signals may appear to be

complex due to presence of imaginary part in the result (imaginary part has negligibly small data values). This should be ignored.

Correlation of two data sequences can be described as summation of the product of two sequences with one sequence shifted, i.e., correletion of $x_1(k)$ and $x_2(k)$ is given as

$$y(n) = \sum_{k=-\infty}^{\infty} x_1(k) * x_2(n+k) \quad ; \qquad - (2.5)$$

In circular correlation, the first signal is multiplied with conjugate of second signal in frequency domain and the result is converted to time domain. Computational error is similar to that given for circular convolution.

3. Interpolation and Decimation . During analog to digital conversion, analog signal is sampled at a fixed rate to get discrete time signal. Subsequently, if there is a need for increase or decrease of sampling rate, Interpolation or Decimation is resorted to. Interpolation (to decrease sampling rate) is governed by interpolation ratio, written as B/A (>1), where A samples are to be interpolated to B samples. For example interpolation ratio of 4/3 means that 3 samples of input signal are interpolated to 4 samples. Following procedure is followed for interpolation of a data sequence:

- Sequence is decimated by picking up every Ath sample.
- In this new sequence now, between each successive data sample B-1 new samples are inserted by interpolation. Turbo Pascal Graphics Procedure 'Splines' (method of cubic splines) is used for interpolation.

To oversample a sequence, Decimation function can be used. Decimation is governed by decimation ratio, written as B/A (<1), where A samples are to be decimated to B samples. The procedure for decimation is similar to that of interpolation.

4. Filtering . Filtering is one of the most common DSP functions. The aim of filtering is to alter the spectral behavior of a signal in a desired form. A filter is defined by its coeffcients. Digital filters are categorised in terms of their impulse response, as FIR or IIR filters .

FIR filters have linear phase characteristics. They typically require many coeffcients and their implementation is same as that of convolution.

FIR filter is defined as

$$y(n) = \sum_{k=0}^{N} b_k \; x(n-k) \; ; \qquad -(2.6)$$

where x(n) is input and y(n) is input. In our implementation, the maximum value of N is 511.

IIR filters have the advantage of sharp frequency cut-off characteristics with relatively low order structure. IIR filter is defined as

$$y(n) = \sum_{k=0}^{N} b_k \; x(n-k) + \sum_{l=1}^{M} a_l \; x(n-l); \qquad \begin{matrix} 0 \le N \le 19 \\ 1 \le M \le 19 \end{matrix} \qquad -(2.7)$$

maximum value of N and M is 19.

5. Windowing . In DSP, a sequence may be having infinite lenth but only a part of waveform is required for analysis or processing, e.g., only one period of a periodic waveform can represent complete waveform. Selection of a part of waveform can be carried out by multiplying it with an appropriate window function. It is felt that the provision of following types of window functions is essential in the software.

(a) Rectangular Window : This window function of lenth N is implemented as

$$W_r(n) = 1 \; ; \quad -(N-1)/2 \le n \le (N-1)/2 \qquad -(2.8)$$
$$W_r(n) = 0 \; ; \quad \text{otherwise}$$

(b) Triangular Window : This window function of lenth N is implemented as

$$W_t(n) = 1 - 2\,n\,/\,(N-1) \; ; \; -(N-1)/2 \le n \le (N-1)/2 \qquad -(2.9)$$
$$W_t(n) = 0 \; ; \quad \text{otherwise}$$

(c) Hanning Window : This window function of lenth N is implemented as

$$W_n(n) = 0.5 + 0.5 \cos(2\pi n / N-1) \quad ; \quad -(N-1)/2 \leq n \leq (N-1)/2 \qquad -(2.10)$$
$$W_n(n) = 0 \quad ; \quad \text{otherwise}$$

(d) Hamming Window : This window function of lenth N is implemented as

$$W_m(n) = 0.54 + 0.46 \cos(2\pi n / N-1) \quad ; \quad -(N-1)/2 \leq n \leq (N-1)/2 \qquad -(2.11)$$
$$W_m(n) = 0 \quad ; \quad \text{otherwise}$$

(e) Blackman Window : This window function of lenth N is implemented as

$$W_b(n) = 0.42 + .5 \cos(2\pi n / N-1) + .08 \cos(4\pi n / N-1) \quad ; \quad -(N-1)/2 \leq n \leq (N-1)/2 \qquad -(2.12)$$
$$W_b(n) = 0 \quad ; \quad \text{otherwise}$$

2.5.2. Arithmetic Functions .

A DSP software should contain all the necessary basic arithmetic fuctions, such as Multiplication, Logarithm etc. These functions provide a facility to the user to carry out required arithmetic manipulations on data sequences. For arithmetic operations such as addition or multiplication, user should have a flexibility of either adding or multiplying two data sequences or one sequence and a constant. Similarly user should be able define base for taking log or exponent of a data sequence. Arihmetic functions provided in the software are designed on these lines only. Additional tools to the user are also provided like conversion of magnitude part of the sequence to decibels and calculation of group delay of the phase part. Following arithmatic functions have been provided.

- **Integration** of real sequence by using simpson rule as given below with x(n) as input and y(n) as output

$$\left.\begin{aligned} y(0) &= 0 \\ y(n) &= y(n-1) + [5x(n-1)+8x(n)-x(n+1)]/12 \\ y(n+1) &= y(n) + [-x(n-1)+8x(n)+5x(n+1)]/12 \end{aligned}\right\} \qquad -(2.13)$$

  where y(n+1) refers to last sample.

- **Differentiation** of real input sequence by calculating the first difference between present and past sample.

- **Addition or Multiplication** of two sequences or one sequence with a real constant.
- **Subtraction** of Second sequence or a real constant from first sequence.
- **Division** of first sequence by a real constant or second sequence.
- **Exponent or log** of a sequence to a specified base.
- Convert real sequence in radians to its **Sine, Cosine or Tangent** values.
- Conversion of magnitude part of data sequence in **Polar form to Decibels**. Magnitude is converted to $20\ Log_{10}$(magnitude).
- Calculation of **Group delay** using phase part of sequence in polar form. For group delay caculation follwing expression is used.

  Group Delay = $-\ (dw\ /\ dx)$; $-$ (2.14)

  i.e., group delay is equal to negative slope of phase change. To calculate this expression where phase part of the sequence is available as digital data, difference between present sample of phase and past sample has been taken as value of group delay at each point. During this calculation, phase difference greater than 2π at any point has been ignored.

2.5.3. Sequence Conversion Functions . The software should contain functions which operate on a sequence to convert it into a different form. This is a facility to the user to manipulate the data sequence to a desired form. Following types of conversion functions have been provided.

- **Rectangular to Polar conversion.**
- **Polar to Rectangular conversion.**
- **Zero Padding** to insert zeroes at desired locations in a real sequence.
- **Extraction** of any part of a real sequence.
- **Delay** of a real sequence in time by inserting leading zeroes.

– Time reversal of a real sequence.

2.5.4. System Response Function .

In a DSP software, which aims to provide study of response of a DSP system, a facility should exist to define the DSP system in many standard ways and to compute time and frequency domain response of the defined system. User may define the system in time domain (difference equation) or in transform domain (transfer function of the system). In transform domain a system may be defined in Direct form, Cascade form, Parallel form or simply by specifying Poles and Zeroes of the system. To study system response in time domain, user can view and store Impulse and Step Response. Similarly frequency domain response is viewed as magnitude and phase parts of the response. More details about types of systems and response are given in following lines:

1. System Description in Time Domain . In time domain, the system is defined in difference equation form.

$$y(n)=\sum_{k=0}^{N} b_k\ x(n-k) + \sum_{l=1}^{M} a_l\ x(n-l); \qquad \begin{matrix} 0 \le N \le 19 \\ 1 \le M \le 19 \end{matrix} \quad - (2.15)$$

where x(n) is the input signal and y(n) is the output signal. System coeffcients b_k and a_l can be in real/complex form. The computation for response of this system is similar to that of direct form.

2. System Description in Transform Domain . In transform time domain, the system is defined in the following standard forms.

(a) Direct form . This form is of the following type.

$$H(Z) = \frac{\sum_{k=0}^{N} b_k Z^{-k}}{1 + \sum_{l=1}^{M} a_l Z^{-l}} \qquad \begin{matrix} 0 \le N \le 19 \\ 1 \le M \le 19 \end{matrix} \quad ; \quad - (2.16)$$

Order of the system will dictate the number of b_k and a_l coeffcients required to

define the system. Between orders of numerator and denominator polynomials, higher order is taken as order of the system. For n order system, user has to feed n times b and a coeffcients.

Time Domain response — Impulse/Step is fed to the system and output obtained.

Frequency Response — FFT of numerator and denominator coeffcients is computed separately. Lenth of FFT is decided by the user as multiple of 2, i.e., 32/64/128 point FFT. Minimum lenth is set at 32 points because display of a smaller lenth does not convey much information. Response output corresponds to 2π in frequency domain.

Frequency Response = FFT of numerator coeffecients / FFT of denominator coeffcients. This division is done pointwise.

(b) Cascade form of second order sections . This form is of the following type (defined by transfer function)

$$H(z) = A * \prod_{k=1}^{N} \frac{1 + b_{k1}Z^{-1} + b_{k2}Z^{-2}}{1 + a_{k1}Z^{-1} + a_{k2}Z^{-2}} \quad ; \; 1 \leq N \leq 8 \qquad - (2.17)$$

where A is a real constant. System can be defined by a_k and b_k coefficients for each cascade section.

Time domain response — Impulse/ Step is fed as input to the first section in cascade and from second section onwards output of previous section is taken as input until final output from the last section in cascade has been obtained. This output is the Impulse/ Step Response of the complete system.

Frequency response — Frequency response H(jw) of each cascade section is computed separately (in a way similar to that given for direct form) and later all such H(jw) values are multiplied together to get frequency response of the complete system.

(c) Second Order Parallel form . Each second order branch is of the following type.

$$\frac{b_{k0} + b_{k1} Z^{-1}}{1 + a_{k1} Z^{-1} + a_{k2} Z^{-2}}$$

System can be defined by a_k and b_k coefficients for each parallel branch.

Time domain response – Impulse/ Step is fed as input to the each parallel branch and all outputs are added together to get Impulse/ Step response of the complete system.

Frequency response – Frequency response H(jw) of each parallel branch is computed separately (in a way similar to that given for direct form) and later all such H(jw) values are added together to get frequency response of the complete system.

(d) Pole-Zero form . A system can be defined by its poles and zeroes. Only real poles/zeroes (i.e., phase angle 0 or 180 degrees) or complex cojugate pole/zero pairs are used to define the system. For real pole/zero, user should feed positive magnitude for 0 degree and negative magnitude for 180 degree phase angle. For complex conjugate pole/zero pair, the values have to be specified in (r,θ) form where only one value of r and θ defines a complex conjugate pair. To compute response of the system, a method similar to that given for direct form has been adopted.

This is a very versatile facility available to the user to be able to define a system by poles and zeroes and study the frequency response. This facility offers to the user a way to experiment by placing poles and zeroes at different locations in the Z-plane and intuitively design a Filter of required specifications.

2.5.5. Quantisation Function .

When a signal is converted from analog to digital, its values are to be converted from infinite precision to finite precision. This effect is similar to adding noise to the signal. Similarly when a filter structure is to be implemented on a computer or on digital hardware, which represents numbers as finite set of binary values, error in accuracy of filter coeffecients occurs. As a result, singularities in Z-plane are permitted to lie only on a few allowable locations. This function in the software provides a facility to quantise a data sequence and coeffcients of a DSP system and to study errors caused due to quantisation.

To simulate a finite precision system, a Quantisation facility is provided, which can be used by specifying number of bits (Ni) for quantisation of integer part and number of bits (Nf) for quantisation of fractional part of the values in infinite precision. In a number 576.753 for quantisation, 576 is integer part and 0.753 is the fractional part.

Quantisation Computation

Let X represent input value to be quantised to Xq value. Range of integer values for quantisation is min = $-2^{(Ni-1)}$ to max = $2^{(Ni-1)} - 2^{-Nf}$.

If X is more than max or less than min, it is scaled to max or min. Let C = 2^{Nf} then for quantisation,

Xq = Round [C X] / C, for X $\geq$ 0

where Round [y] rounds off y to the nearest integer. In a sequence if any sample value is beyond the limits of min or max, the complete sequence is scaled down with a scale factor (min/value or max/value) before quantisation and scaled up after quantisation by reverse scaling factor.

For example, Let Ni=5, Nf=3, then C =2^3 =8, min=-16, max=15.875

- for X =3.1415926, Xq = Round [8*3.1415926]/ 8 = 25/8 = 3.125

– for X = – 37, scaled X = –16, Xq = Round [8* (-16)] /8 =–16

and after scaling back Xq = –37.

Sequence and Coeffcients quantisation

Quantisation function offers quantisation facilities for a data sequence and coeffcients of a system defined in direct form or cascade form of second order sections. These system types are same as explained in Para 2(a) and 2(b) of 2.5.3. Frequency response of the system is computed to show the effects of quantisation. Computation of frequency response is similar to that given in Para 2(a) and 2(b) of 2.5.3 Details about display of output can be read from Appendix-B.

2.6 DISPLAY MODE

Display facility enables the user to view the data stored in any buffer at any time. The display facility always first shows the buffer status with information about occupancy state of all four buffers and type of DSP operation carried out on data before its storage. For example buffer status is displayed as

– Buffer 1 contains output of FFT operation.

– Buffer 2 is empty.

– Buffer 3 contains output of MULTIPLICATION operation.

– Buffer 4 is empty.

During display of data, the type of data stored (rectangular/ polar), is also indicated to the user by appropriate headings. Display of buffer data is possible by graph plot or tabulation.

1. Tabulation of Data . A maximum of 20 points of data lenth can be tabulated at a time on the screen. A heading indicates type of data and lenth of data is indicated by an index.

2. Graph-Plot of Data . Turbo Graphics Toolbox has been used for plotting of graphs in a graphics window. Ref Appendix-B for more details.

2.7 OUTPUT MODE

Disk files are used for storage and printing of data. Data stored in a disk file can be used at any time later for further processing or study of output of a DSP operation. A DSP software has to be fully interactive with disk files for transfer of data.

2.7.1 Facilites offered by Output mode

In DSP software, output mode can be entered for writing of data to a file. The source of data can be of any type available in standard input menu options. Thus the output mode offers a large variety of options as given below.

1. By selecting disk-file as source of data, one can transfer data, without leaving the software, from one file (in any drive / directory) to another file in any drive / directory.
2. By selecting buffer as source of data, result of any DSP operation stored in a buffer, can be transferred to a file and that buffer can be re-used for for storage of data.
3. By selecting any standard library function (e.g., Impulse, Sine etc.) as source of data, a signal can be generated and stored in a file for later use.

CHAPTER - 3

SYSTEM SIMULATION

2.1 INTRODUCTION

The software facility discussed in chapter 2 provides a fairly comprehensive and flexible signal processing environment to create signals, simulate algorithms and display results. In practical applications, there arises a need to simulate a complex DSP system consisting of a sequence of signal processing operations. In DSP.Exe, user can implement one function at a time, starting from front end, store the result in a buffer and use it as input for the next function to be called. In this manner, a fairly complex DSP system can be simulated and studied. However, the extreme user interaction provided by DSP.Exe may be quite distracting and time consuming especially when the user is implementing a long chain of DSP functions to simulate a complex DSP system. In such applications, it is quite unlikely that the user would want to examine the output at each node. Instead one would like to simulate a part or the complete system at one go without much interruption. User may possibly want to examine the output at a few selected nodes during part or complete run of the system. In such cases, DSP.Exe will warrant much unnecessary interaction from the user and will thus prove to be cumbersome and time consuming. In the context of system simulation, the user typically defines the system in a block diagram flow graph, specifies the block parameters and then runs the system. To meet such requirements in system

simulation applications, another software SYSTEM.Exe has been developed which overcomes the drawback encountered in DSP.Exe when used as a tool for complex system simulation. It allows the complete system to be defined at the begining before the part/complete system is run by the user. If required, the system description information can be stored in a file for subsequent retrieval and re-use. Editing of block parameters is possible to change parameters later if such a need arises. In this manner a complex system once defined need not be defined again other than editing of desired parameters. The software also provides a 'Display During Run' facility in addition to grph-plot, tabulation and file storage of the current result. In the following sections we discuss the design approach in developing the software and the various facilities provided. The user manual for SYSTEM.Exe is given in Appendix-C.

2.2 SYSTEM DESIGN CONSIDERATIONS

The DSP system can be defined as a number of functional blocks, each block performing a signal processing task. These blocks are interconnected in such a way that output data of each block goes as input to some other block, so as to perform a desired sequence of operations. In a DSP chain, the first block has to be input generation block and elsewhere in the chain also these blocks may appear as and when input generation is required. Rest of the blocks can be defined as blocks performing a signal processing, mathematical or sequence conversion function. Hence, to define and run a DSP system, the user has to follow the following steps.

1. Define the system in terms of number of functional blocks and provide function code for each block.

2. Provide relevant block parameters for each block (e.g., filter coefficients for a filter block).

3. View system parameters by Edit facility to check correctness of parameters and make amendments, if any, in the parameters.

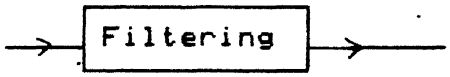
4. Run part/complete system as desired. During the run, user can also see display of intermediate results by specifying the nodes before run is executed.

5. Study the result by graph-plot/tabulation or store the result in a file.

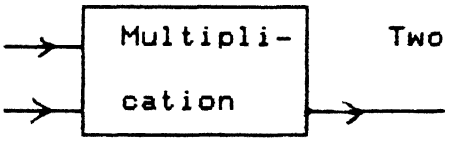
2.2.1 Types of Functional Blocks

Each functional block has one or two interconnections for taking input and one interconnection for output, other than input generation block which has only output interconnection. The examples of all types of blocks are as given below.

No input interconnection, only output interconnection.



Single input and output interconnections.



Two inputs and one output interconnections.

The blocks can be categorised under the main headings of Input, Signal processing, Arithmatic and Sequence conversion functions. The details of these functions have been discussed in Chapter 2 under Input Mode and Function Mode. Each function

has been assigned a function code for ease of the user in defining a block (as given in Table C.1 of Appendix-C). There are a total of 25 types of functional blocks provided as explained in following lines.

1. Input Block . Function code – 0. In a DSP system, the first block has to be input block for generation of input signal. At any other place in the system, whenever a block does not take input from previous block but needs an input either from standard input library or from a disk file, an input block has to be inserted before this block. Standard library functions offered in input block are :

- Impulse
- Step
- Sine
- Cosine

2. Signal Processing Blocks . Function codes – 1 to 10.

3. Arithmatic Blocks . Function codes – 11 to 16.

4. Sequence Conversion Blocks . Function codes – 17 to 24.

2.2.2 Buffers for Computations

After a DSP system has been defined and is to be run, software buffers are required for storage of intermediate data. As we have seen in the example for type of blocks in 2.2.1, a block requires a maximum of two input interconnections. Considering a block with two inputs (e.g. , correlation, addition etc.), if both the inputs are also coming out as output of two blocks previously executed, it means that we need two independent buffers for storing the result of these two previous blocks. From these buffers, stored data is read as input for execution of later block with two input interconnections. If the system defined does not have any block with two inputs, only one of the buffers may be used for storing the output data of last block executed and will supply input data for next block to be

executed. In such a case, the second buffer remains unused.

SYSTEM.Exe file uses two such buffers for data storage of intermediate results, namely buffer A and B. The characterstics of these buffers are similar to that explained in Section 2.3 of Chapter 2. During run of the system, following sequence of action is adopted for storage and retrieval of intermediate results.

- Buffer A is filled up with generated data sequence from first input block.
- If at any time, another input block is to be executed, Buffer B is filled up with generated data sequence.
- Blocks are executed one by one, taking input from previous blocks as per interconnections defined by the user.
- Output of every block is stored in the same buffer from where it had taken input. The only exceptions are the blocks with two inputs whose output always goes to buffer A.
- The output of last block executed is always in buffer A, from where it can be used for display or storage in a disk-file.

Guidelines for making of block diagram by the user to define a DSP system are explained in Appendix-C. These guidelines help the user to define the system in a way to suit the sequence explained above and to avoid any errors in computations.

2.3 EDITING OF SYSTEM PARAMETERS

Once a system has been defined in terms of number of blocks, function code for each block and relevant parameters for each block, it is possible to view and edit the parameters of the system. This is an essential facility to ensure correctness of system parameters before executing the system. Also, after a run of the system has taken place, user may not be satisfied with the result and may want to alter

system parameters to achieve the desired result.

This facility offers a special feature when the system parameters have been read from a file where they were stored by the user during an earlier use of the software. In this case, user is not required to re-define a system which has been defined and tested earlier, yet he can make amendments in the system parameters to carry out desired alterations.

2.4 RUN OF THE SYSTEM

During run of the system, it is possible to run upto any block, by starting always from first block. The facility to run part of the system is essential in case intermediate results are required to be displayed by graph-plot/tabulation or stored in a file. During run, the execution of blocks always begins from the first block. This method of running the system may be slightly time consuming but it avoids occurence of user mistakes, like running the system from an intermediate point without running the previous portion of the system.

During run of the system, it is possible to display output of any block being executed by 'Display During Run' facility. To provide intermediate displays, the system is run upto the specified block, output of this block is displyed to the user and then rest of the system is run. As explained in Section 2.2.3, intermediate results are not stored in the buffer but the output of the last block, executed in a run, is stored. This output can be studied by graph-plot or tabulation of data. The last result can also be transferred to a disk file for storage. All these facilities are similar to those explained in detail in Chapter 2.

CHAPTER - 4

CONCLUSIONS AND SCOPE FOR FURTHER WORK

In the previous chapters, a DSP software has been discussed which offers the user an environment for signal processing and simulation of complex DSP systems. The software is a complete package by itself and functions independently on an IBM compatible PC with an hard disk and a Hercules/CGA/EGA graphics card.

The project had begun with an ambition to fulfil a number of aims which have been largely achieved but some work still remains because of paucity of time. There remains, therefore, some scope for further work to improve upon and add more facilities to the existing software. Some of these aspirations are discussed in the following sections.

4.1 Addition of More User Functions

Though the software offers a large number of functions to operate on digital data, more functions can be added to create an extensive library. These functions may be designed to perform matrix/vector operations which are required in a number of signal processing applications.

4.2 Enhancement of Graphics

The existing graphics can be improved in the following areas to make the software more interactive and presentable.

– Labeling of axes during graph-plot.

- Marking of axes at appropriate places (e.g., mark $\pi/2$, π etc on x axis for display in frequency domain).
- Plotting of multiple graphs on one screen.
- Plotting of more than one graph in a number of windows on one screen.

4.3 Filter Design

Filter design facility is an integral part of a DSP software. There is a requirement to provide an executable file to design a filter and store filter coeffcients in a file from where they can be read by DSP.Exe during implementation of filters. A number of design methods are available for filters. Some of the references, dealing with filter design/programs, are being mentioned below.

1. 'DSP Algorithms' by Bateman and Yates.
2. 'Digital Filter Design' by T. W. Parks and C. S. Burrus.
3. 'Signal Processing Algorithms', by Samuel D. Stearns and Ruth A. David, Prentice Hall, 1988.
4. 'Digital Signal Analysis', by Samuel D. Stearns and Ruth A. David, Prentice Hall, 1990.
5. 'Handbook for DSP', by D.F. Elliot, Academic Press, 1987.
6. 'Digital Filter Design Handbook', by F. J. Taylor, Dekker 1983.
7. 'Programs for DSP' , by IEEE DSP Committee, 1979.
8. 'Discrete Time Signal Processing', by A. V. Oppenheim and R. V. Schafer, Prentice Hall, 1989.
9. 'Digital Filters and Signal Processing', by L. B. Jackson, Kluwar, 1988.
10. 'Digital Signal Processing', by D. J. Defatta, Wiley, 1988.
11. 'Introduction to DSP', by J. G. Proakis and P. G. Manolakis, Macmillan, 1988.

4.4 Complex Waveform Generation

Complex Waveform routines in a DSP software offer an extensive library of various types of signals for study and processing. Some of the desirable waveforms to be included are listed below.

1. Amplitude modulation and Frequency Modulation.
2. Amplitude Sweep and Frequency Sweep.
3. Frequency Shift keying.
4. Sinc Pulse.
5. Random Number Generators. etc.

Such waveforms find extensive use in complex communication system simulation. Standard routines in Fortran for generation of above mentioned waveforms are available in 'Arbitrary Waveform Cookbook' by WAVETEK.(Wavetek, San Diego Inc, USA, 1988)

4.5 Waveform Measurement

It is desirable to have waveform measurement capabilities such as statistical and pulse parameter measurements. Some of these facilities to be included are given below.

Pulse Parameters Measurement – Amplitude, Rise time, Fall time, Duty factor, Positive and negative peaks and Period of the pulse.

Statistical Parameters –Mean, Standard Deviation, Variance and RMS value of a waveform.

4.6 Acquiring Waveform Data

The present software provides facilities to generate data sequence and to read data from a disk file. The facility to acquire data as an data arrary output of an instrument will be an attractive feature. For this, software routines for I/O have to be added to the software, to interact with with PC ports, together with an

Anolog to Digital card as hardware support. Similarly, to send out data to an anolog instrument, a Digital to Analog card has to be provided with the PC. To interact with I/O ports, data sequence is stored in a file before sending an output data or after acquiring an input data.

4.7 Real time Data Processing

This is the future enhancement of the present DSP software to implement a DSP system in real time, by acquiring data on line, processing the data and sending the processed data to an analog destination. For speeding up the data acquisition and processing, the software should be converted to C or Assembly Language to interact faster with the hardware. The associated hardware support has also to be provided. For more details in this direction, user should refer to [7] and [8].

REFERENCES

[1] M. Fashano and A. L. Strodtbeck, ' Communication System Simulation and Analysis with SYSTID', IEEE J. on Selected Areas in Communication, vol. SAC-2, No. 1, pp. 8-29, Jan. 84.

[2] W. H. Tranter and C. R. Ryan, ' Simulation of Communication System using Personal Computers', IEEE J. on Selected Areas in Communication, vol. SAC-2, No. 1, pp. 13-23 Jan. 88.

[3] Technical Manual for 'MATLAB'.

[4] Malcolm Slaney, ' Interactive Signal Processing Document', IEEE ASSP Magazine , vol. 7, No. 2, pp. 8-20, Apr. 90.

[5] Tektronix Manual, 'Signal Processing and Display', Tektronix Inc., 1988.

[6] Matti Karjalainen 'DSP Software Integration by Object Oriented programming : A Case Study of Quicksig', IEEE ASSP Magazine , vol. 7, No. 2, pp. 21-31, Apr. 90.

[7] Alan Kamas and Edward A. Lee, 'Digital Signal Processing Experiments', Prentice Hall, 1989.

[8] Jeffrey C. Bier et.al.,' Gabriel: A Design Environment for DSP ', IEEE Micro, pp. 28-45, Oct. 90.

[9] David G. Messerschmitt, 'A Tool for Structured Functional Simulation', IEEE J. on Selected Areas in Communication, vol. SAC-2, No. 1, pp. 137-147, Jan. 84.

[10] Andrew Bateman and Warren Yates, ' Digital Signal Processing Design', Pitman Publishing, 1988.

[11] Alan V. Oppeinheim and Ronald W. Schafer, ' Digital Signal Processing ', Prentice Hall, 1988.

[12] Alan V. Oppeinheim and Ronald W. Schafer, ' Discrete Time Systems ', Prentice Hall, 1989.

[13] Roman Kuc, ' Introduction to Digital Signal Processing ', Macgraw hill, 1988.

[14] Samual D. Stearns, ' Digital Signal Analysis ', Prentice Hall, 1990.

APPENDIX - A

SOFTWARE INFORMATION

A.1 THE DSP SOFTWARE

This DSP package is a digital signal processing software package designed to offer DSP facilities on a PC with hard disk. The software has been written using Turbo Pascal version 5 and graphics facilities have been provided using Turbo Pascal Graphics Toolbox. There are a total of 13 files in the package (as listed in A.2) out of which two are executable files, namely DSP.Exe and SYSTEM.Exe, which can be called from DOS for signal processing and system simulation works respectively. DSP.Ovr is the overlay file for DSP.Exe. DSPIT.Tpu and DSPIT1.Tpu files hold a library of subroutines being used by the two exe files. DSP.Exe requires both tpu files whereas SYSTEM.Exe requires only DSPIT.Tpu file. Rest of the files listed in A.2 are from Turbo Pascal Graphics Toolbox to support graphics requirement of the software.

A.2 DETAILS OF SOFTWARE FILES

There are 13 files in the package as given below.

1. DSP EXE – Signal processing software.
2. DSP OVR – Overlay file for DSP.Exe.
3. SYSTEM EXE – System simulation software.
4. DSPIT TPU – Library of subroutines.
5. DSPIT1 TPU – Library of subroutines.
6. GRAPH1 TPU – Modified Gdriver.Tpu.
7. GRAPH2 TPU – Modified Gkernel.Tpu.
8. GRAPH3 TPU – Modified Gshell.Tpu.
9. GRAPH4 TPU – Modified Gwindow.Tpu.

10. 14*9 FON – Turbo pascal font file.
11. 8*8 FON – Turbo pascal font file.
12. 6*4 FON – Turbo pascal font file.
13. ERROR MSG – Turbo pascal error message file.

A.3 INSTALLATION OF THE SOFTWARE

The software has been developed for an IBM compatible PC with a hard disk. It is possible to run system simulation part from a floppy drive also by loading files in serial 3,4 and 6 to 13 (total 10 files) in the floppy. The software is compiled on an XT with Hercules Graphics card. However, the software can be installed on a PC with CGA OR EGA also. For such a requirement, differnt version of software has to be used with graph files for relevant graphics card. These versions are available with thesis supervisor. For more details about graphics, user can refer to Turbo Pascal Graphics Toolbox manual.

APPENDIX - B

DSP - USER MANUAL

B.1 MAIN MENU

DSP software can be run by calling DSP.Exe file from DOS.

B.1.1 User Interaction

1. Modes . When DSP.Exe file is run , a main menu appears on the screen. It offers following modes.

⟨I⟩ Press I to enter Input mode (Sec B.2).

⟨F⟩ Press F to enter Functions mode (Sec B.3).

⟨D⟩ Press D to enter Display mode (Sec B.4).

⟨O⟩ Press O to enter Output mode (Sec B.5).

⟨Home⟩ Press Home key to enter Help mode (Para 2).

⟨End⟩ Press End key to quit Para 3).

When any of these options is selected by the user, a sub menu appears in a graphic window, which offers further options. When user has finally decided on a given option, main menu disappears from the screen.

2. Help . Help mode is effective as long as it is displayed on the screen. It may be entered by pressing Home key available on the keyboard. When help is called, detailed information about the software is displayed on the screen ,which extends into number of pages. After user escapes from help display ,by pressing End key ,he returns to the begining of main menu.

3. To Escape from current location . End key, available on the keyboard, can be

used for escaping from anywhere in the menu or sub-menu. When user escapes from the sub-menu he returns to the begining of main menu. If user escapes from main menu ,he returns to DOS. Escape option is possible as long as "End -Quit" is displayed on the screen.

B.1.2 Buffer

Buffers are used for temporary storage of data, after performing a DSP task.

1. Lenth of Buffers . Four working buffers are provided to the user, each of maximum lenth upto 512 points. User can decide lenth of each buffer for storage or retrieval of data. The data in each buffer is stored as real and imaginary parts usually, unless data is in polar form (where magnitude and phase parts are stored).

2. Selection of Buffer . After a DSP function is run by the user, he is given a choice to store resultant data in one of the four buffers. If user decides to store data and selects a buffer already holding data, he gets a warning about the buffer being full. Now user has a choice to either overwrite data previously stored or select another buffer.

3. Buffer Memory . Each time a buffer is used for storage of data, the lenth of data stored is recorded. Whenever user wishes to read data from a buffer (Sec B.2.1), recorded lenth of data stored is displayed to the user and user is given a choice for selection of data lenth for reading. If data lenth in buffer is M and while reading the buffer user selects a data lenth N, then

(a) If M < N then the read data is automatically zero padded by N−M trailing zeroes.

(b) If M > N then only first N data values are read from the buffer.

4. Display of Buffer Contents . User can view the contents of any buffer at any time by selecting Tabulate or Graph-Plot option in Disply mode. In Display mode,

headings appear on top of gaphics window/table, which inform the user about type of data being displayed, i.e., Rectangular or Polar.

5. Data Storage in Disk-Files . Whenever user falls short of storage space due to the restriction of maximum number of working buffers being four, he should store data in disk-files by entering Output mode and selecting Writefile option. Any number of files can be used for storage and retrieval of data. Thus a combination of buffers and disk-files provide a large working space to the user.

B.1.3 Examples to Illustrate Use of Buffers

1. Reading and Storage of data .

- User enters Input mode / enters Functions mode to select a DSP function.
- User is prompted to select input type for current operation from standard input menu. If buffer is selected as input type, user can decide on lenth of data to be read (Para 1 of sec B.1.2).
- After the specified operation is performed, result is displayed by graph-plot and user is given a choice to store resultant data in a buffer. At this time buffer status is also displayed.
- User may select an empty buffer for storage of resultant data.

2. Display of Buffer data .

- User enters Display mode and selects one of the filled buffers for display.
- Buffer data is displayed by Graph-plot / Tabulation.

3. Transfer of Buffer to Disk-File .

- User enters Output mode for file writing where he is prompted to select source of data from standard input menu. User selects buffer as source of data and gives specific buffer number from where data is transferred to file.

B.2 INPUT MODE

Input mode can be entered from main menu. When entered, it displays a number of options available, for selections of input type, from a standard input menu in a graphics window. After a selected input is generated, user is given a choice for storing it in a buffer from where it may be used at any time, as required by the user. In Function mode, when a DSP function is called by the user, he is prompted to select input type for current DSP operation from the same standard input menu.An input buffer of maximum lenth 512 points is filled up with selected input type data of real/complex nature.

B.2.1 Input Options

User can select one of the following types of signals from standard input menu.

⟨B⟩ Press B to select buffer data as input. Buffer status is displayed to the user. User can select a non empty buffer as source of input. Lenth of input data to be read can be upto a maximum of 512 points (Para3 of Sec B.1.2).

⟨F⟩ Press F to select disk-file as input. If file is not in current DOS directory, complete path should be specified while giving file name.

⟨K⟩ Press K to select keyboard input option. After specifying lenth of input in points, user can feed real or complex data values directly from keyboard.

⟨I⟩ Press I to select impulse as input signal.

⟨P⟩ Press P to select pulse as input signal.

⟨S⟩ Press S to select sine as input signal.

⟨C⟩ Press C to select cosine as input signal.

⟨D⟩ Press D to select dc as input signal.

⟨R⟩ Press R to select Ramp as input signal.

B.2.2 Parameters for input options .

If selected input type is chosen from last five options (impulse to ramp), following input parameters are required to generate the signal.

1. Amplitude in volts.
2. Offset from origin in points - positive or negative.
3. Width of pulse in points - only for pulse signal.
4. Lenth/Period in points - upto a maximum of 512 points.
5. Extend Periodically - Input signal, as defined by parameters in serial 1 to 4, can be extended periodically upto a maximum of 512 points of input buffer lenth. For a non periodic signal of lenth N, (N (512), later 512-N points of input buffer are filled with zeroes.

B.2.3 Display of selected input signal

After an input signal has been specified, as per Sec B.2.1 and B.2.2, the signal is generated and input buffer is filled up. The generated signal is automatically displayed by graph-plot. For complex signal, real and imginary parts are displayed separately. After display, if Input mode was entered from main menu, the generated signal can be stored in a working buffer for later use. On the other hand, if standard input menu was called for selection of input type for a DSP function in Functions mode, this selected signal is used as input to the DSP function.

B.3 FUNCTION MODE

Function mode can be entered from main menu. When entered, it displays a number of functions submenu options in a graphics window. The options are listed below.

⟨S⟩ Press S to select Signal Processing option.

⟨A⟩ Press A to select Arithmatic option.

⟨C⟩ Press C to select Conversions option.

⟨R⟩ Press R to select System Response option.

⟨Q⟩ Press Q to select Quantisation option.

Details about these options are given from Sec B.3.2 to B.3.6.

B.3.1 Sequence of action in Functions mode

Every function option prompts the user to select input(s) by displaying 'Select Input Type' for current operation, as explained in Sec B.2. If the called function requires mare than one input, user is prompted twice to select first and second inputs (as in the case of convolution, addition etc). The selected input signal is automatically displayed by graph-plot.

After selection of input signal(s), user is asked to supply relevant parameters of the function concerned (e.g., for FFT function user provides number of FFT points). Now, the function is executed and the result is displayed by graph-plot. The result is in rectangular or polar form depending upon type of function executed. For real result, only real part is displayed. If the result is complex, real and imaginary parts (for rectangular result) or magnitude and phase parts (for polar result) are displayed separately.

B.3.2 Signal Processing Functions

Following functions are available in signal processing.

1. FFT . Input type : Real / Complex.

This function computes Fast Fourier Transform of input signal. The output of this function is always complex, in rectangular form. Depending on lenth of input signal, next higher lenth in power of 2 is displayed as suggested lenth for FFT computation. User can choose either the suggested lenth or any lenth (in power of

2) upto a maximum of 512 points. The lenth of output corresponds to 2Π radians in frequency domain.

2. Inv FFT . Input type : Complex.

This function cumputes inverse Fast Fourier Transform of input signal. The output of this function is always complex, in rectangular form. Lenth selection for Inv FFT computation is similar to that given for FFT. The lenth of input corresponds to 2Π radians in frequency domain.

3. DFT . Input type : Real / Complex.

This function cumputes Discrete Fourier Transform of input signal. The output of this function is always complex, in rectangular form. If input signal is of lenth L, any lenth N (2 ≤ N ≤512) can be chosen for computing the DFT sequence. If N>L then input is automatically zero padded with (N−L) zeroes. The user interaction is similar to that of FFT.The lenth of output corresponds to 2Π radians in frequency domain. Since DFT computation is a slow process, FFT should be used for long input signals.

4. Inv DFT . Input type : Complex.

This function cumputes Inv Discrete Fourier Transform of input signal. The output of this function is always complex, in rectangular form. Lenth selection for Inv DFT computation is similar to that given for DFT. The lenth of input corresponds to 2Π radians in frequency domain.

5. Convolution . Input type : Two signals of Real / Complex nature.

This function convolves the two input signals. User is prompted to select two input signals from standard input menu and then he is given a choice between following two types of convolution processes.

(a) Convolution by computation in time domain . This is a slow process as lenthy time domain calculations are involved. Output is real for both real input signals and complex for both complex input signals.

(b) Fast(Circular) Convolution . This is a fast process, since both signals are multiplied in frequency domain and result is converted to time domain. FFT is used for conversions between time and frequency domains. If input signals have lenths of N1 and N2 points, the output lenth will be next higher power of 2 to N1+N2−1 points. The output is always complex.

6. Correlation . Input type : Two signals of Real / Complex nature.

This function computes cross correlation of the two input signals. For auto correlation both signals should be same. The user interaction is similar to that of convolution.

7. Interpolation . Input Type : Real.

This function interpolates and adds an integer number of points between selected number of points of input signal. User is required to select A and B, where number of input samples A are interpolated to B samples, with interpolation ratio B/A>1. The lenth of output is greater than lenth of input depending upon interpolation ratio.

8. Decimation . Input Type : Real.

This function decimates the input signal to an output signal as per decimation ratio specified by the user. User is required to select A and B, where number of input samples A are decimated to B samples, with decimation ratio B/A<1. The lenth of output is less than lenth of input depending upon decimation ratio.

9. Filtering . Input Type : Real / Complex.

(a) FIR . This function runs the input signal through Finite Impulse Response filter whose coefficients are specified by the user. The coefficients can be fed in real/complex form directly from the keyboard or read from a file. FIR filter is defined as

$$y(n) = \sum_{k=0}^{N} b_k \; x(n-k) \; ; \quad 0 \leq N \leq K \qquad - (B.1)$$

where k=511 for file input of coeffcients and k=19 for keyboard input of coeffcients. x(n) is input and y(n) is input.

b) IIR . This function runs the input signal through Infinite Impulse Response filter whose coefficients are specified by the user. The coefficients are to be fed in real/complex form directly from the keyboard. IIR filter is defined as

$$y(n) = \sum_{k=0}^{N} b_k \; x(n-k) + \sum_{l=1}^{M} a_l \; x(n-1); \qquad 0 \leq N \leq 19$$

$$1 \leq M \leq 19 \qquad - (B.2)$$

10. Windowing . Input Type : Real / Complex.

This function multiplies input signal by a selected window. User is required to specify type of window, its lenth and offset from origin. Following types of window functions can be selected.

- Rectangular window
- Triangular window
- Hamming window
- Hanning window
- Blackman window

B.3.3 Arithmatic Functions

The functions under this sub-menu carry out mathematical manipulations on input signal(s). Following functions are available.

1. Addition . Input Type : Two signals both Real / both Complex .

This function pointwise adds an input signal with a real constant or another signal. If the lenths of signal are unequal, the shorter signal is zero padded to make both signals of equal lenth.

2. Subtraction . Input Type : Two signals both Real /both Complex .

This function subtracts a real constant or second input signal from first input

signal. If the lenths of signal are unequal, the shorter signal is zero padded.

3. Multiplication . Input Type : Two signals both Real / both complex.

This function pointwise multiplies an input signal with a real constant or another input signal. If the lenths of signal are unequal, the shorter signal is zero padded to make both signals of equal lenth.

4. Division . Input Type : Two signals both Real / both Complex .

This function pointwise divides first input signal by a real constant or another input signal. Second input must not contain any zero valued samples or an error will occur due to division by zero. Therefore the second signal must not be shorter in lenth than the first signal.

5. Integration . Input Type : Real.

This function estimates the integral of the input signal.

6. Differentiation . Input Type : Real.

This function computes first derivative of input signal by calculating difference between present and following sample. The output lenth is shorter than input lenth by one point.

7. Minimum . Input Type : Real.

This function finds the minimum value and its location in an input signal. If same minimum value exists at more than one location (e.g., in periodic sequence), the first location is found.

8. Maximum . Input Type : Real.

This function finds the maximum value and its location in an input signal in a manner similar to finding minimum.

9. Logrithm . Input Type : Real and Positive.

This function computes log (pointwise) of input signal to the base specified by the user.

10. Exponent . Input Type : Real and Positive.

This function computes exponent (pointwise) of input signal to the base specified by the user.

11. Mag in Db . Input Type : Real / Complex (in polar form).

This function converts (pointwise) magnitude part of input signal into decibels.

12. Sine . Input Type : Real (in Radians).

This functions poitwise computes Sine of input signal.

13. Cosine . Similar to Sine.

14. Tangent . Similar to Sine.

15. Group Delay . Input Type : Complex (in polar form).

This function computes group delay of phase of phase part of input signal.

B.3.4 Sequence Conversion Functions

The functions under this sub-menu operate on input data sequence to alter it to desired form. Following functions are available.

1. Zero pad . Input Type : Real.

This function pads input signal with zeroes at specified locations. User is required to specify start and end points for zero padding. After zero padding, input signal from start point onwards is reproduced, hence no sample of input signal is lost. The output signal is longer than input signal depending on number of zeroes added.

2. Extract . Input Type : Real.

This function provides a facility for extraction of a part of the input signal. User is required to specify start and end points for the purpose of extraction. The output sequence contains the extracted part.

3. Delay . Input Type : Real.

This function delays a time domain input signal by a specified number of samples. The output signal has leading zeroes corresponding to delay lenth.

4. Reversal . Input Type : Real.

This function reverses a time domain input signal.

5. Rectangular to Polar . Input Type : Complex (rectangular form).

This function converts a rectangular input signal (i.e., real and imaginary parts) to polar output signal (i.e., magnitude and phase parts).

6. Polar to Rectangular . Input Type : Complex (polar form).

This function is inverse of 'Rectangular to Polar' function.

B.3.5 System Response

This function facilitates the user to define a DSP system, by specifying system coeffcients in a number of standard ways, and study the system response in form of Impulse/Step/Frequency response. After seeing display of the response of a particular system, user can store current result in a buffer and again calculate response for the same system or define a new system for study.

B.3.5.1 Types of DSP Systems . A DSP system can be defined in the following ways.

1. System Description in Time Domain . In time domain, the system is defined in difference equation form.

$$y(n) = \sum_{k=0}^{N} b_k \; x(n-k) + \sum_{l=1}^{M} a_l \; x(n-l); \qquad 0 \le N \le 19, \quad 1 \le M \le 19 \qquad - (B.3)$$

where x(n) is the input signal and y(n) is the output signal. User is required to provide b_k and a_l coefficients in real/complex form.

2. System Description in Transform Domain . In transform time domain, the system is defined in the following standard forms.

(a) Cascade form of second order sections . This form is of the following type.

$$H(z) = A * \prod_{k=1}^{N} \frac{1 + b_{k1} Z^{-1} + b_{k2} Z^{-2}}{1 + a_{k1} Z^{-1} + a_{k2} Z^{-2}} ; \quad 1 \leq N \leq 8 \qquad - (B.4)$$

where A is a real constant. User is required to feed real a and b coefficients for each cascade section, upto a maximum of 8 cascade sections.

(b) Direct form . This form is of the following type.

$$H(Z) = \frac{\sum_{k=0}^{N} b_k Z^{-k}}{1 + \sum_{l=1}^{M} a_l Z^{-l}} ; \quad 0 \leq N \leq 19, \quad 1 \leq M \leq 19 \qquad - (B.5)$$

User has to specify order of polynomial before feeding b_k and a_l coeffecients. Between numerator and denominator polynomials, higher order should be selected as order of polynomial of the system. For n order system, user has to feed n times b and a coeffecients.

(c) Second Order Parallel form . Each second order kth branch is of the following type.

$$\frac{b_{k0} + b_{k1} Z^{-1}}{1 + a_{k1} Z^{-1} + a_{k2} Z^{-2}} \qquad - (B.6)$$

User has to feed b and a coeffcients for each second order parallel branch, upto a maximum of 10 parallel branches.

(d) Pole-Zero form . A system can be defined by its poles and zeroes. A restriction here is that only real poles/zeroes (i.e., phase angle 0 or 180 degrees) or complex cojugate pole/zero pairs can be used to define a system. For real pole/zero, user should feed positive mgnitude for 0 degree and negative magnitude for 180 degree phase angle. For complex conjugate pole/zero pair, the values have to be specified in (r,θ) form where only one value of r and θ defines the complex conjugate pair.

B.3.5.2 Response of the DSP System . Response of the system, as defined by B.3.5.1 can be studied in one of the following ways.

1. Impulse Response . User has to specify lenth of response computation upto a maximum of 128 points.

2. Step Response . Similar to Impulse response.

3. Frequency Response . Frequency response is computed by using FFT. User has to specify lenth of response as 32/64/128 points.

B.3.5.3 Display For System Response . After computation of system response, the result is automatically displayed by graph-plot. The lenth of display is already specified by the user as lenth of response in B.3.4.2. If system was defined in cascade form, user can see response of each cascade section before seeing the final response of the complete system. For Impulse/Step response of the system, only real output is displayed whereas for frequency response user can see magnitue,phase and group delay of the response result.

B.3.6 Quantisation

Quantisation sub-menu provides following options.

B.3.6.1 Input Quantisation . Input Type : Real.

This function quantises the input signal, to the level specified by the user. User is required to specify number of bits for quantisation of integer part and fractional part separately for sample values of input signal. The number of levels for quantisation is 2^n , where n is number of bits specified for quantisation. The output is quantised input which is automatically displayed by graph-plot. Quantised output is plotted in broken line along with input signal in continuous line. User can see the graph for quantisation error also.

B.3.6.2 Coeffecient Quantisation . This function quantises the coeffecients of a system defined by the user, computes frequency response of the system with unquantised and quantised coeffcients and displays the comparative responses by graph-plot. User interaction for number of bits is similar to that of Input Quantisation. Output of this function is frequency response of the system with quantised coeffcients. To define a system, coeffcients can be specified in following two ways.

1. Cascade form of second order sections . This form is of the type as given in B.3.5.1. User is required to feed real a_k and b_k coefficients for each cascade section, upto a maximum of 10 cascade sections.

2. Direct form . This form is of the type as given in B.3.5.1. User has to specify order of polynomial before feeding b_k and a_k coeffcients. Between numerator and denominator polynomials, higher order should be selected as order of polynomial of the system. For n order system, user has to feed n times b and a coeffcients.

Display and Storage of Frequency Response . Frequency response is computed by using FFT. User has to specify lenth of response as 32/64/128 points. After computation of frquency response, the result is automatically displayed by graph-plot. Response of the system with quantised coeffcients is plotted in broken line along with response of system with unquantised coeffcients in continuous line. Magnitude and phase parts of response are displayed separately. Frequency response of the system with quantised coefficients is available as output for storage in a buffer.

B.4 DISPLAY MODE

Display mode can be entered from main menu. It is a facility to view the contents of the buffers at any time. It offers following options.

⟨T⟩ Press T to tabulate data of selected buffer. Data lenth on each screen is of 20 samples.

⟨D⟩ Press D to Graph-plot data of selected buffer.

B.4.1 Graph-Plot of Buffer Contents

Graph-plot facility has been provided by Turbo Pascal Graphics Toolbox. In a graph, scaling of axes is always from 0.0 to 9.99. To plot values above or below this limit, a multiplication factor appears ($10^1, 10^2, 10^3 \ldots$ or $10^{-1}, 10^{-2}, 10^{-3} \ldots$) on extreme right side of x-axis and on top of y-axis. To read the graph points, the multiplication factor must be kept in mind. Scaling on x-axis represents sample points and scaling on y-axis represents amplitude of sample points. If an error message ('Too few points—plotting not possible') appears while trying to groph-plot a buffer, it indicates that data lenth of buffer is smaller than 2 points.

B.5 OUTPUT MODE

Output mode can be entered from main menu. It is a facility available to transfer data to a disk file. User is prompted to select source of data from standard input menu. The selected sequence is automatically displayed by graph-plot. User can utilise output mode in one of the following ways.

1. Select source of data as file from any drive / directory to be transferred to a new file in any drive / directory.

2. Select source of data as buffer (holding result of a DSP operation) to store data in a file.

3. Select source of data from standard input library functions (Impulse, Sine etc.), to generate desired signal and store in a disk-file.

APPENDIX - C

USER MANUAL -SYSTEM SIMULATION

System simulation software can be run by calling SYSTEM.Exe file from DOS.

C.1 MAIN MENU

When SYSTEM.Exe file is run, a main menu appears on the screen which offers following options.

1. Define new system.
2. Edit system parameters.
3. Run system (part/complete).
4. Graph plot for output.
5. Tabulate output.
6. Store output in file.
7. Store system parameters in file.
8. Quit to DOS.

The first option for the user is to define a new system.. Options 2 to 7 always refer to the current system defined by the user. After operations pertaining to any option is performed, user automatically returns to main menu from where he can select same or any other option.

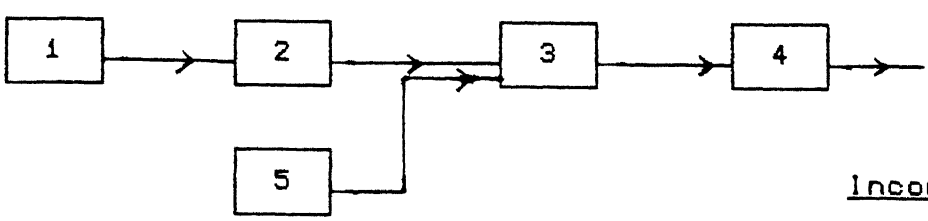
C.2 DEFINE NEW SYSTEM

Before selecting a new system, user should prepare a block diagram for reference, with each signal processing operation defined by a block (see example in C.2.2). To define a new system, user is first required to specify the number of blocks in the system and function code for each block. Each DSP operation is assigned a separate function code (as shown in Table C.1). These codes are displayed on the right side of the screen for the ready reference of the user for defining a fuctional block type. User is required to select codes from code table and feed one code for each block. The first block is assigned code 0 (Input block) by default.

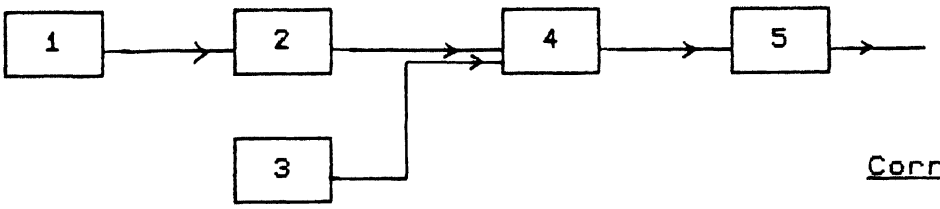
The number of blocks that can be used to define a system is restricted to 10. If user wishes to define a system longer than 10 blocks, he should break his system into two or more systems, with the lenth of first system as 10 blocks and the lenths of subsequent systems upto a maximum of 10 blocks for each system. User has to now define and run each system one by one . To execute this type of system, user has to define and run each system starting from first, store each result in a file and then define and run subsequent system which takes input from the same file. This technique can be adopted for a system of any lenth.

C.2.1 Guidelines for making Block Diagram (ref 2.2.2 for details)

1. Blocks should be numbered for the purpose of reference. The numbering should be from left to right, starting from 1 and going upto 10 or less.

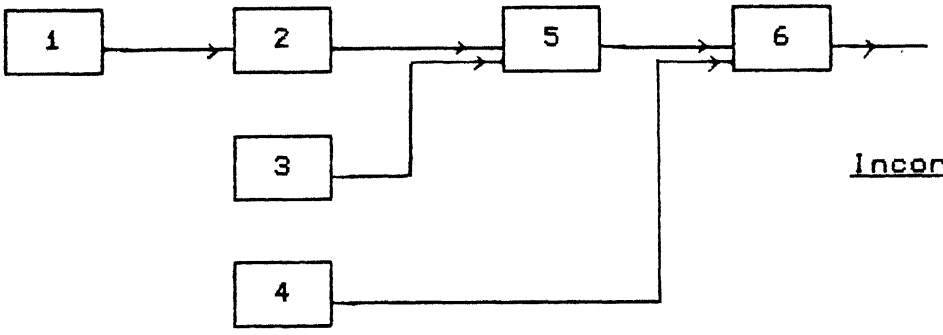


Incorrect Numbering

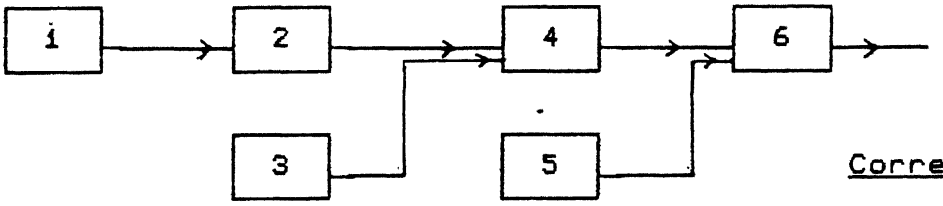


Correct Numbering

2. At any place, more than two parallel blocks should not be created.

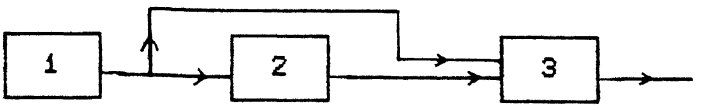


Incorrect Design

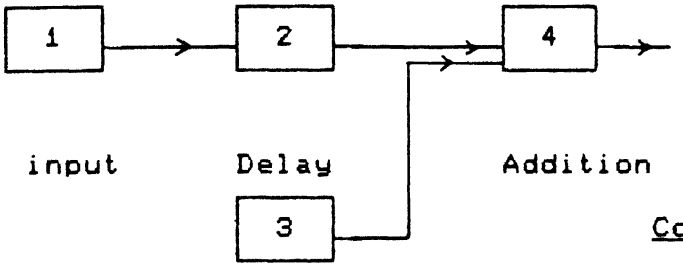


Correct Design

3. No feed forward of output of any block should take place. If such a requirement arises, a duplicate parallel chain should be created of similar blocks. This constraint is due to limited buffer memory which can not store all intermediate results.



Input Delay Addition Incorrect Feed Forward



Correct Design

(parallel Input Block Created)

Input(duplicate of block 1)

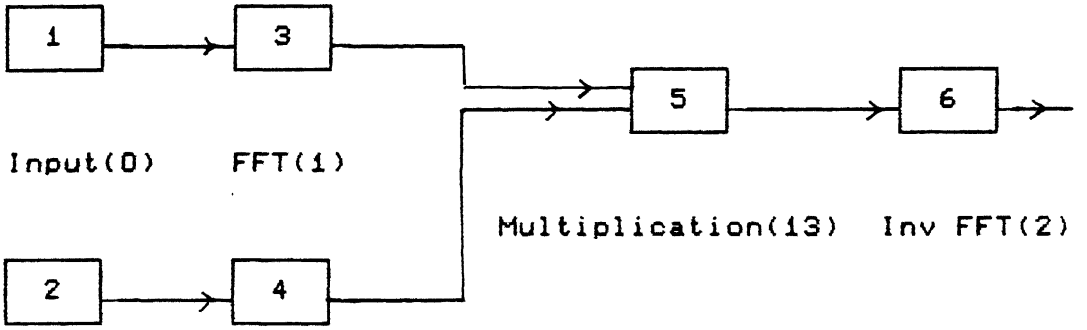
4. No feed back interconnections are possible in the present implementation.

C.2.2 System Parameters

When the system has been defined in terms of number of blocks and function code for each block, user is required to feed parameters for each block and specify interconnections of blocks. Parameters are asked from the user depending upon fuction code of each block.

Example

To perform fast convolution of two input signals.



Note – Numbers in the brackets reprsent function code of each block.

An illustration of parameters is given in the following lines (underlined values are parameters as defined by the user).

BLOCK-1 :Input

Input type: Input from file - a:\data\in.dat

(similar parameters for block 2)

BLOCK-3 :FFT

Input selection : Enter n (output of nth block as input)

Select input : 1

How many points FFT (4/8/16/32/64/128/256/512) : 64

(similar parameters for block 4)

BLOCK-5 :Multiplication

Input selection : Enter n (output of nth block as input)

Select input-1 : 3

Select input-2 : 4

Above example illustrates all types of blocks available for system simulation ,i.e.,

Blocks 1 and 2 — Input generation blocks (No preceding interconnctions, only input parameters required).

Blocks 3 and 4 — FFT blocks (single input interconnection, FFT parameters required in terms of FFT points).

block 5 — Multiplication block (Two input interconnections, no parameters required).

C.3 EDIT SYSTEM PARAMETERS

From the main menu edit facility can be called to complete system parameters in a tabular form and to make changes in parameters of any block if so desired. It is

recommended that user should view system parameters before running the system, so as to avoid run-time errors and errors in output due incorrect parameters. If an error occurs during run, an appropriate error message is displayed. During edit, all parameters of the a block can be edited, other than redefining the block itself by changing its function code.

C.4 RUN SYSTEM

From the main menu run facility can be called to execute the system currently defined. User can run part/complete system by starting from block 1. During run, if user wishes to see any intermediate/last result by graph- plot, he should specify the block(s) number for 'Display During Run'. When the system is being executed, output of specified block(s) is displayed before the succeeding block is run. After each display, if user feels that the result is not proper he can abort the subsequent run.

Any part of the system can be run any number of times starting from first block. After the run, the 'current output' is available for graph-plot/tabulation/storage. By current output, we refer to the output of last block executed in the last run. Therefore every important result should be first stored in a file before run is called again.

C.5 GRAPH PLOT FOR OUTPUT

After run of part/complete system is executed, the current output can be seen by graph plot facility available from main menu. If the system has not been run after

being defined, graph plot is not possible since no output data is available.

As display of output is possible during run also, this facility has been incorporated in the main menu to enable the user to display the current output again at any time after run execution is over.

C.6 TABULATE OUTPUT

Tabulation of current output offers a facility similar to that of graph-plot as explained in C.5. Tabulation of 20 samples of data at a time is possible on the screen.

C.7 STORE OUTPUT IN FILE

This option can be called from main menu to store current output data in a disk file. While specifying name of the file, complete path should be mentioned otherwise current directory is selected by default for creation of file.

C.8 STORE SYSTEM PARAMETERS IN FILE

Current system parameters can be stored in a disk-file. Later, while selecting a new system (Sec C.2.1 and C.2.2), these parameters can be read from the file to select the same system. Now, this system can be used as such or after editing of desired parameters.

Table C.1

FUNCTION CODES

Type Of Fuction	Function code
Input	0
FFT	1
Inv FFT	2
DFT	3
Inv DFT	4
Interpolation	5
Decimation	6
Windowing	7
Filtering	8
Convolution	9
Correlation	10
Addition	11
Subtraction	12
Multiplication	13
Division	14
Integration	15
Differentiation	16
Rectangular to Polar	17
Polar to Rectangular	18
Logrithm	19
Exponent	20
Delay	21
Zero Pad	22
Extract	23
Sequence Inversion	24